Central Control Over Distributed Routing

Why Fibbing?

This paper focuses on intra-domain link state routing. While SDN and Openflow provide better routing manageability by allowing fine grained central control over routing decisions, the same central nature sacrifices the robustness of distributed protocols and results in adhoc scalability. Fibbing is an architecture that achieves both flexibility and robustness through central control over distributed routing algorithms such as OSPF and IS-IS.

SDN is getting traction recently and edge networks are migrating to SDN but there is already a huge installed base of OSPF speaking routers, management tools and human operators that are not familiar with the technology so Fibbing can work as an intermediate step towards an all-SDN future. Some motivating reasons elaborated in the paper are given below:

• Congestion in a certain part of network caused by sudden surge of traffic requires network management actions such as isolating suspicious traffic, redirection to a scrubber for inspection and cleaning and load balancing on under-used links to alleviate congestion. To accomplish these tasks, current traffic engineering protocols like MPLS RSVP-TE would need to configure many tunnels and hence would increase control-plane and data-plane overhead. SDN can achieve these configurations but in a network of hundreds of nodes, SDN would not be able to as quickly adapt to topology changes as a distributed protocol would do given the fact that rule updating by a central controller would become a bottleneck. Fibbing on the other hand can steer flows to any set of per-destination paths via underlying distributed routing protocol so Fibbing readily supports traffic engineering, load balancing, fast failover and traffic steering through middleboxes in a manageable and distributed fashion.

• Fibbing is as scalable as the underlying distributed routing protocol, since Fibbing leverages link state routing protocol’s signaling for control-plane communication. In case of controller failure, Fibbing downgrades to default OSPF functionality.

• Fibbing works on existing routers where routing entry installation time is less than 1ms. This offers much faster convergence than is possible with current SDN/Openflow implementations.

Although Fibbing offers attractive features of scalability and robustness when compared with traditional SDN, considering the context of comparison is important. Traditional SDN is a whole package providing fine grained control over traffic as well as measurement capabilities. Fibbing
on the other hand is specific to forwarding applications which is only a subset of what SDN offers.

**System Design**

In a traditional OSPF network, routers compute shortest paths over a synchronized view of the topology. Fibbing supports advanced forwarding applications by changing the topology view for routers. Fibbing inverts the routing function; given the desired routing entries, Fibbing computes the topology view that would result in desired routing entries. More specifically, Fibbing influences routers into computing desired forwarding entries by presenting them with a carefully constructed augmented topology that includes fake nodes providing fake announcements of destinations and fake links with fake weights. Fibbing provides its own high level language to operators to specify policies (forwarding requirements). The language supports combining various requirements as well as wild cards.

Fibbing controller performs three tasks

- Generate fake augmented topology using forwarding policies and network topology.
- Establish link state adjacencies with routers in network to learn topology and inject announcements about fake nodes.
- Track network changes and their impact on augmented topology to react and update topology.

Fibbing workflow consists of following steps

- Topology Initialization: Controller modifies weights in link state protocol to support topology augmentation by Fibbing.
- Controller generates augmented topology in two passes. The first pass generates topology to meet requirements (per-destination augmentation), the second pass optimizes the topology by reducing fake nodes (cross-destination optimization) and consequently control and data-plane overhead.

**Implementation and Evaluation**

Since Fibbing controller needs to interact with other routers (speak OSPF) so it is implemented by extending Quagga. OSPF’s *Forwarding Address* field is used to introduce fake node announcements. The evaluation section is sparing in terms of experiments and over simplistic in terms of assumptions made. Two sets of experiments were performed

- **Router Measurements**: These experiments were performed on a single commercial router (Cisco ASR9K, 12G DRAM).
  1. Effect on memory and CPU with fake nodes: As number of fake nodes are increased from 1000 to 100,000, RIB memory increases from 0.09MB to 39.78MB and OSPF memory increases from 0.56 to 113.17MB.
2. Routing entry installation time for a single entry remained constant as number of fake nodes were increased. This result however does not take into account network dynamics and considers only the time needed to install a particular entry at router.

3. Routing protocol convergence time remained same even after injection of fake nodes. This result again considers a scenario where network dynamics are ignored and router’s convergence time is considered more like a stand alone entity.

- **Topology Augmentation Evaluation:** This controller side experiment evaluates how quickly controller adapts to changing topology and was performed offline by simulation on realistic ISP topologies from Rocketfuel. Forwarding requirements were generated by randomly changing next hops of randomly selected nodes. Experiments with varying number of fake nodes indicate that Fibbing computes augmented topology within ms. Also cross-destination optimization reduces number of fake nodes on average by 25%.